Real-World Product Deployment of Adaptive Push Notification Scheduling on Smartphones

Tadashi Okoshi
slash@sfc.keio.ac.jp
Keio University

Kota Tsubouchi
ktsubouch@yahoo-corp.jp
Yahoo Japan Corporation

Hideyuki Tokuda
hxt@sfc.keio.ac.jp
Keio University

ABSTRACT

The limited attentional resource of users is a bottleneck to delivery of push notifications in today's mobile and ubiquitous computing environments. Adaptive mobile notification scheduling, which detects opportune timings based on mobile sensing and machine learning, has been proposed as a way of alleviating this problem. However, it is still not clear if such adaptive notifications are effective in a large-scale product deployment with real-world situations and configurations, such as users' context changes, personalized content in notifications, and sudden external factors that users commonly experience (such as breaking news). In this paper, we construct a new interruptibility estimation and adaptive notification scheduling with redesigned technical components. From the deploy study of the system to the real product stack of Yahoo! JAPAN Android application and evaluation with 382,518 users for 28 days, we confirmed several significant results, including the maximum 60.7% increase in the users' click rate, 10 times more gain[1]compared to the previous system, significantly better gain in the personalized notification content, and unexpectedly better performance in a situation with exceptional breaking news notifications. With these results, the proposed system has officially been deployed and enabled to all the users of Yahoo! JAPAN product environment where more than 10 million Android app users are enjoying its benefit.

CCS CONCEPTS

• **Human-centered computing** → **Smartphones**; *Ubiquitous and mobile computing systems and tools.*

KEYWORDS

interruption overload, push notification, attention management, smartphone, middleware, real-world deployment

ACM Reference Format:
Tadashi Okoshi, Kota Tsubouchi, and Hideyuki Tokuda. 2019. Real-World Product Deployment of Adaptive Push Notification Scheduling on Smartphones. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330732

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330732

1 INTRODUCTION

Limited attention resource of human is a big bottleneck to delivery of push interruptive notifications in today's computing environments [8] where users receive large amounts of various information on mobile and wearable devices. This issue has recently come into the spotlight as the numbers of interruptive push notifications reaching users' smartphones have drastically increased.

In such "interruption overload" situations, many researchers [2, 7, 11, 27, 28] have been working on improving the mobile user's receptivity to information while preserving the limited human attention resource by using various types of sensing, machine learning, and data science technologies. Several systems have been proposed for detecting opportune (i.e., interruptible) timings for delivery of notifications [24, 25, 27, 28]. Evaluations of such systems (including user studies and analysis of collected data from sensors and human behaviors) have revealed that delivering notifications at such timings (detected via mobile sensing and machine learning) is effective in terms of lowering the user's cognitive load and mental effort, shortening response times, and raising click rates. Recently, evaluation of such adaptive notification in the product environment was conducted for the first time [26] and resulted in a significantly lower click response time as long as higher click rate and higher engagement level to the web services.

However, in spite of the findings above, towards the permanent deployment of this technology we have the following signifncant challenges. (1) It is still not clear if such adaptive notification system can result significantly better performance (than the conventional notification system) on the real product environment. In the previous evaluation [26], the gain of users' notification click[2]brought by the introduction of adaptive scheduling in notification delivery was less than 2%, which is not convincing enough for us to make a business decision to deploy this system permanently to our infrastructure. According to our previous analysis [36], one cause is found to be the differences of users' physical behavior between weekdays and weekend days, leading major performance degradation in ML-based interruptibility classification. This is obviously one of the possible performance improvement opportunities for the system. Furthermore, it is still not clear if our adaptive notification has applicability to several types of the real-world configurations and situations around notifications, such as (2) customized notification content prepared especially for each individual user, and (3) sudden external factors that users commonly experience (such as breaking news).

[2]A comparison of the performance values before and after introducing the proposed method, calculated by division of both numbers.
E.g., $gain(click\_rate)$ = $click\_rate(with\_our\_method)/click\_rate(without\_our\_method)$

In this paper, we address these three challenges and reports the performance of our new adaptive notification system deployed on the real product environment, with the real users, the real notification contents, and the real situations. Our large-scale deployment of the system with more than 380,000 real users for 28 days revealed the followings. On the first challenge, as the highlight of the results, our system achieved more than 10 times more gain (23.3% on average and 41.6% at maximum) in the end user's click-through, compared with the gain of in the previous system (1.9% gain). On the second challenge, with our adaptive scheduling, we found that notifications with personalized content had more than double the gain in click-through rate compared with the case with generic contents. On the final challenge, interestingly, we found that our system resulted in even greater gain during a day when numerous push notifications on unexpected breaking news (which were outside our system's scope) were issued. With these significant results, our adaptive scheduling system has officially been deployed to Yahoo! JAPAN's product environment as a permanent component, and currently used by more than 10 million Android application end users. To the best of our knowledge, this work is the first to report such extensive evaluation and the permanent deployment of an adaptive notification in a real product environment.

The remainder of this paper is organized as follows. Section 2 explains the interruption overload problem in mobile computing. Section 3 discusses related work. Section 4 clarifies our research challenges and research approaches to address them. Section 5 presents the system design and architecture of our system. Section 6 reports on the overview of our large-scale deployment study with 382,518 users. Section 7, 8, and 9 present the results and analysis. Section 10 discusses research opportunities arising from our study. Section 11 concludes this paper.

2 MOBILE USERS AND INTERRUPTION OVERLOAD

In today's mobile and ubiquitous computing environments, users are starting to suffer from **interruption overload** by large numbers of interruptive notifications presented in inappropriate ways. Interruption overload can be treated as a sub class of the broader information overload problem discussed for many years [22, 31, 35]. In particular, a number of studies have been conducted in the context of interruptions and multitasking [1, 10, 29, 36–38].

The main source of interruptions is notifications of various types, such as system status updates from local operating systems, messages from other users, breaking news, and other information from applications and services. The concept of notification in computer systems originated from the desire to provide the latest information to the user in a more timely and speedy manner than by periodic polling. As more and more applications and services run in the background (both local and remote) of the user's current foreground task, and as typical notification systems deliver incoming notifications immediately due to the original system design intention above, users must now face numerous interruptive notifications at random timings, regardless of what they are doing as their primary task. When a notification is perceived and recognized by a user, some amount of his/her limited attention [19, 30] is allocated to the presented information. When an interruption occurs, this situation is called "divided attention" [33].

Past studies have revealed several negative influences of interruptive notifications, such as on productivity [3, 4, 6, 20, 32, 39], emotional and social attribution [3], and psycho-physiological states [39]. Interruptions also increase the time needed for the user to restart his/her primary task after attending to a notification [3]. One obvious fallback strategy of the user is to disable the notifications either completely or temporarily. However, this negates the benefits of notifications. It is reported that users actually prefer to keep using notification systems even with their interruption costs, rather than disabling them and polling new information manually [18].

The need for computing systems that can adapt their behavior to the limited attentional resources of users has been gradually recognized, as more and more literature particularly on sensing and estimating the user's current attentional states has appeared.

3 RELATED WORK

Although we have already introduced some of the past literature as background on our research, we describe here those studies most directly related to our work.

In contrast to the above methodologies that try to measure the level of cognitive load relatively directly, detection of a user's **interruptibility**, in relation to the sources of possible interruptions, has recently been proposed as a relatively indirect and abstract scale for measuring the user's current attentional states. This class of research can be mainly categorized into two different groups: (a) estimation of interruptibility at certain timings based on the various contexts of the user and (b) detecting **breakpoints** [23], i.e., the boundary between two adjacent actions of the user. In particular, breakpoints are found to be the times at which interruptions cause the user less frustration and cognitive overhead [3, 15, 16].

The early research on interruptibility [5, 12–14, 17] was limited to the desktop computing environment, with more recent studies examining it in relation to mobile and ubiquitous computing. Ho [11] found that interruptions at activity transitions can reduce annoyance in situations where users wear wireless accelerometers. Fischer [7] found that users tend to be more responsive to notifications at breakpoints after phone calls and text messages than at other random timings. Hofte [34] used an experience sampling methodology (ESM) to collect information on location, transit status, company, and activities to build a model of interruptibility, particularly for phone calls. Pejovic expanded the use of context to detect interruptible moments on smartphones, including user activity, location, time of day, and emotional states [27].

Recent studies have even considered user boredom [28] as yet another opportunity for delivering notifications. At the system level, the current situation of fragmented interruptive notification delivery over mobile networks is inefficient and power hungry. Acer [2] showed that delaying delivery of notifications can yield power savings in mobile devices.

In our previous studies [24, 25], we focused on how user's breakpoints can be detected in real time on commodity smart and wearable devices (e.g., smartphones and watches) without a dedicated psycho-physiological sensor, and how scheduling notification deliveries affects the user's response. Furthermore, we conducted the first evaluation [26] in a real-world product environment where the interruptibility estimation logic was embedded in a commercial

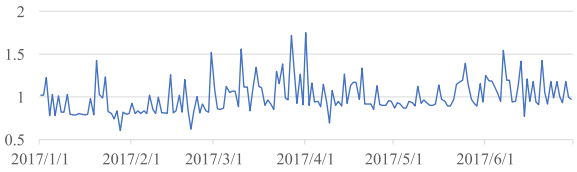

Figure 1: Daily Total Number of Notifications Sent on Yahoo! JAPAN Android Application (1Q 2017)

Yahoo! JAPAN Android application. That evaluation revealed that delivering notifications at such detected timings is beneficial to users and showed significantly quicker response times, as well as more clicks and higher levels of engagement with the service.

4 CHALLENGES AND APPROACHES

4.1 Challenges Towards Product Deployment

In spite of the existing work presented above, we still foresee significant challenges to understand how adaptive notifications contribute to the behaviors and experience of users in the real world, towards permanent deployment of this technology to our product infrastructure.

C1. Insufficient performance gain in changes in user activity: Previous system [26] resulted in 1.9% click gain (denoted by $gain = click\_rate(withsystem)/click\_rate(withoutsystem)$) compared in a evaluation in the product environment. Considering the additional component both on the client side (embedded in the application) and on the server side (including a Hadoop cluster resource), this number was not convincing enough for us to be confident of introducing this technology as a permanent component in the product stack. According to our data analysis [36], the largest contributing features in the breakpoint detection model on weekdays were significantly different from those on other days (Saturdays, Sundays and national holidays), and this looks to be a cause of degradation and instability in the ML classification performance for interruptibility estimation.

C2. System's performance in handling personalized notifications: In the real world, most notification services send content that is personalized for each user or a group of users. For example, many services push out recommending content (very often including advertisements) for users with particular interests (e.g., "finance" or "football") or demographic attributes (e.g., "male of the age between 30 to 39"). The previous research did not investigate the performance of adaptive notification scheduling in a real-world product environment with personalized notifications. Investigating how well the system works with such practical notifications remains a challenge.

C3. System's effectiveness during occasional notification spikes: Figure 1 shows the total number of notifications sent daily from the Yahoo! JAPAN Android application (to all users) during the first quarter of 2017. (The numbers in the graph are ratios with 1.0 being the average during this period; the absolute value is confidential.) The plot shows occasional spikes where the daily number of notifications are obviously higher than that of the previous day. According to our analysis, these spikes come from breaking news notifications of occurrences of earthquakes, typhoons, and threats from other countries. Such breaking news notifications are sent to all users of the app immediately regardless of their preferences. Because of their nature as emergency messages, such notifications cannot be delayed and must be sent promptly. We should then ask ourselves, even in such a real-world situation with the occasional spikes, how well does adaptive notification scheduling behave (for non-emergency notifications) in terms of the click rate and users' response time.

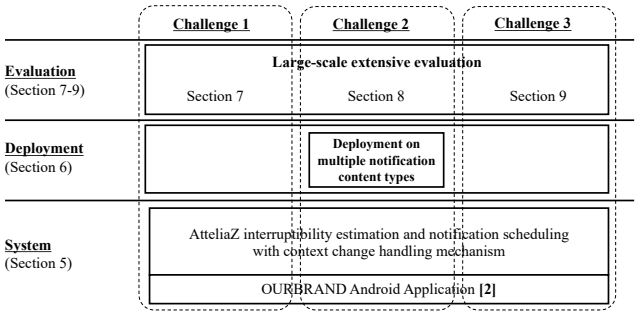

Figure 2: Challenges and Our Approaches to Meeting Them

4.2 Approaches

In this research, we aim to address the above challenges with the approaches illustrated in Figure 2. For C1, we construct a new mechanisms of interruptibility estimation along with a corresponding adaptive notification scheduling on the system layer (Section 5), with vairious new technical compornents suchas new features and the interruptiblity estimation classifier. For C2, we deployed our system to handle two different types of notification classes to see how well our mechanism works in case of personalized notifications (Section 6). On top of these, our large-scale study reports the evaluation results for C1 (Section 7), C2 (Section 8), and C3 (Section 9) respectively.

5 THE SYSTEM

Our system for Yahoo! JAPAN Android application consists of the basic architecture that we have built [26] and various component redesigned specifically for this study, such as new features for capturing users' contexts and machine learning algorithms.

5.1 The Architecture

Our approach to interruptibility estimation is to detect each user's breakpoints [23] (boundaries of two adjacent activities), which are considered to be opportune moments for users to receive information because they are periods of lower cognitive load and mental effort [24, 25]. Our system illustrated in Figure 3 detects such timings during the user's physical activity on a smartphone in real time, by using mobile sensing (Google's activity recognition [9] for the user's current activities) and machine learning techniques. Using a stream of activity labels along with other sensor readings on the device, features are calculated, and a trained linear regression model identifies whether the current moment is a breakpoint. Once a notification is received on the phone, the corresponding adaptive notification component delays its presentation to the user until the next breakpoint is detected.

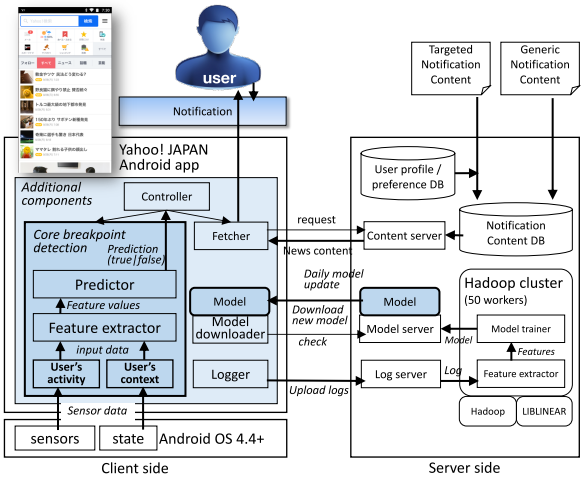

Figure 3: System Architecture

Table 1: Features Extracted From The Sensor Data

Event Types	Explanation	Levels
Timestamps (Hour)	Hour value extracted from time stamp of client	24
Collected sensor data	Present status of the client device	36
Trigger	Sensor type triggering this detection process	3
Transitions of sensor values	"From and To" transition sets of each sensor activity recognition (6x6), charging(2x2) application volume(15x15), system volume(9x9), silent mode(4x4), network(6x6)	474

The system automatically presents a notification after a specified timeout T if no breakpoint is detected. On the basis of empirical knowledge, we set this period to 1 hour. The user's reaction to the notification, along with the current sensor data, is logged on the client. When a user clicks a notification within 10 seconds of the breakpoint that triggered the notification, all the sensor data at that breakpoint timing is logged together with the ground-truth annotation, "breakpoint". The system sends the annotated log data to the server periodically. Every night on the server side, a new model is built from all the data uploaded from the clients and sent to all the clients. (Note that this system does not have a personalized model for breakpoint detection; thus, all users use the same model updated by the server nightly.)

5.2 Handling user context changes from one day to the next

For our first challenge of better system performance and stability to changes in user activity, in this research we developed and deployed new machine learning components on top of the baseline system [26].

5.2.1 Features. In order to capture users' activity more precisely, additional features were implemented in our classification model compared with the baseline system. The total number of features was 474, as shown in Table 1.

5.2.2 New Classifier Design. The evaluation of previous systems revealed two ways to improve it in terms of the design and creation of the classification model. Firstly, with the daily model update scheme in which a classifier model is trained on the server side every night and is distributed to all the users in a daily-basis, sometimes the model update failed for reasons such as temporary network failures. Since the two models (weekdays and weekends) tended to be significantly different from each other [36], executing an unmatched model may lower performance. Thus, a single classification model would ideally be preferred to refer to differences in the day of the week and other varying contextual situations of the users. Thus, in this study, we redesigned the classifier model by introducing context variations so that single model can accommodate such differences. Table 2 lists the new contexts introduced in the redesigned classifier.

Table 2: Additional Context in Classifier

Context Name	Values
Day of the week	Weekdays, Saturday, Sunday, Holiday
Time of day	1AM-6AM, 7AM-noon, 1PM-6PM, 7PM-24AM
Vibration mode	on, off
Device sleep	on, off

First, we introduced new features that account for the current context value in a Kronecker product. In the previous system, linear regression was used to optimize the weight vector w.

$$y = \sum_i w_i \cdot x_i + w_0 \quad \ldots (1)$$

where x_i denotes the ith feature and w_i denotes its weight.

When introducing a binary context value, {Weekdays, Weekends} for example, we split the features in the product as follows.

$$y = \sum_i w_{d,i} \cdot x_{d,i} + \sum_i w_{e,i} \cdot x_{e,i} + w_0 \quad \ldots (2)$$

where $w_{d,i}$ is the weight for weekdays and $w_{e,i}$ is the weight for weekends. (In the actual system, we split up the product even further with more contexts.)

Secondly, even with a single model that can accommodate such versatile contexts, one has to consider unbalanced amounts of ground truth for each contextual situation. For example, there is much less ground truth data available on weekends (Saturday and Sunday) compared with on weekdays (Monday to Friday). Poorer performance can be expected for rare contextual situations until enough ground truth for them is obtained. Hence, we added another term to the weight calculation, as follows.

$$y = \sum_i w_{t,i} \cdot x_{t,i} + \sum_i w_{d,i} \cdot x_{d,i} + \sum_i w_{e,i} \cdot x_{e,i} + w_0 \quad \ldots (3)$$

where additional $w_{t,i}$ denotes the weight of the ith feature not considering any contextual situation. In this way, such additional weights will be heavily utilized in the early stage of the daily learning with the new system. As the learning iterations add more ground truth data even for such rare situations, context-aware weights, such as $w_{d,i}$ $w_{e,i}$, are gradually activated and the performance for rare situations eventually improves.

6 DEPLOYMENT STUDY

Towards the final goal of the system's permanent deployment into the real product environment of Yahoo! Android application, we firstly deployed the system for a subset (but still a large number) of the users for 28 days to validate its performance.

Table 3: Notification Classes

Class Name	Real-Time / Batched	Content Examples
Breaking news	real-time	breaking news articles
Natural disaster	real-time	weather update, earthquakes
Recommendation	batched	sports & showbiz news, tips
Personal	batched	transit, travel, targeted content

6.1 Target Users

382,518 users of the Yahoo! JAPAN Android application (male: 60.6%, female: 39.4%) were recruited as participants in this evaluation. We built a new test infrastructure on our mobile client and cloud server, wherein we could enable or disable a specific functional feature for a specific class of user on the basis of their device IDs. We randomly split the selected users into (a) an experimental group (users to which our mechanisms of real-time breakpoint detection and notification delivery at those timings were enabled) and (b) a control group (users to which our logic was not activated). The selection was not visible to the participants. The application release was checked and confirmed by the corporate legal and compliance departments. The study was conducted in conformity to the corporation's regulations and end user agreement.

6.2 Duration

The evaluation was conducted for about one month in the summer of 2017. The overall effective evaluation duration was 28 days. Since release of a new version of the Android application takes approximately 3 to 5 days to reach all users in our company's gradual deployment scheme for preventing large-scale service problems, the evaluation period needed an initial deployment period at the beginning and it is not included in the 28-day effective measurement period.

6.3 Initial Model Training

The initial model used in the beginning of this deployment is what we trained throughout our previous user study [26]. For more detail, please refer our previous work. Although we added more features in our new system (compared with the system previously constructed), since we are using linear regression, the weight parameters from the old models can be used with several zero weight values for the additional features.

6.4 Configuration on Notification Content

Table 3 shows the different notification classes currently used in Yahoo! JAPAN services. Particularly for our challenge (2), we enabled the system with two classes, "Recommendation" and "Personal", to evaluate how well our system works for different types of notification.

6.4.1 Generic Notification Content. The "Recommendation" notifications mainly consisted of non-emergency updates from Yahoo!services, mostly on sports and showbiz-related news. They did not include any emergency content that needed to be presented to users immediately, i.e., not deferred via breakpoint detection. All users who enabled reception of this class on the app's preference screen received exactly the same content. The notifications had four daily delivery timing slots (8AM, noon, 6PM, and 9PM). The content to be delivered was posted in a batch queue by the news provider department and sent to each user at the scheduled timings. Only one notification was pushed to the users at each timing slot.

Since every user received the same content in the same timing slot regardless his/her preferences or interests, we could use this notification class as the baseline for investigating the user's receptivity without having to worry about influence from personalization aspects of the notification content.

6.4.2 Targeted Notification Content. The "Personal" notification class included several different notification types, including ones specific to each user (i.e., calendar update, travel update) and ones specific to groups of users with certain interests (e.g., football game updates for those who have "football" in the interest field of their profiles.) The latter sub-class of notifications is called "Targeted" in Yahoo!, and we decided to deploy the system on this type of notification. Targeted notifications do not have fixed delivery timing slots, although often (but not always) the news sender intentionally sends out notifications at the so-called "golden time" (e.g., 9PM) according to their best practices. After the content is posted by the news provider department, our back-end system finds a preferable group of users and searches its log information; then it sends out the notification to those users.

Although this class of notification is not actually 100% "personalized" for each user, we can still use it to evaluate the effectiveness of the system for notifications with a certain degree of content-interest optimization.

6.5 Measurements

Various user manipulation logs inside the Android application, including timestamps of when content arrives at the device and when a notification is clicked by the user, are periodically recorded along with sensor values and uploaded to the server. As described in the system design, the sensor data was annotated with "breakpoint" ground truth when the user clicked a notification within 10 seconds, as described in Section 5.1. This data was used in our analysis and for the nightly model training.

6.6 Collected Data

During the study period, the server sent the total number of 77 generic and 261 targeted notifications to each user. We collected several terabytes of log data in total from all the clients. The amount of ground truth data used in the daily model training was approximately 15GB (about 36 million lines of data).

7 RESULT (1): GAIN IN USERS' CLICK RATE

First, we focus on the user's click rate for presented notifications. Recall that the click gain is defined as $gain = click\_rate(with\_adaptive\_notification\_scheduling) \,/\, click\_rate(without\_adaptive\_notification\_scheduling)$. Also, please note that we use relative "gain" values in this paper since the absolute click rate cannot be disclosed to the public. (It is corporate confidential. We internally discussed about possible disclosure of the absolute numbers but unfortunately could not change the business-oriented decision.)

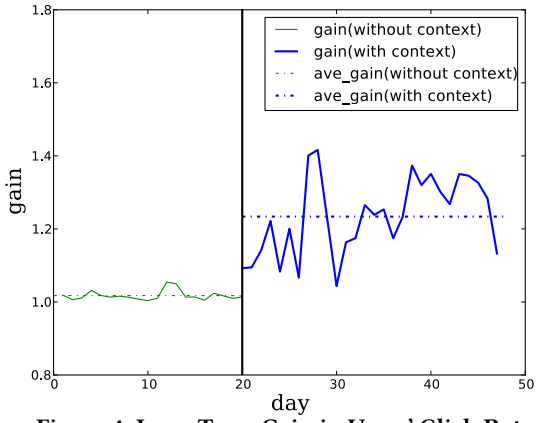

Figure 4: Long-Term Gain in Users' Click Rate

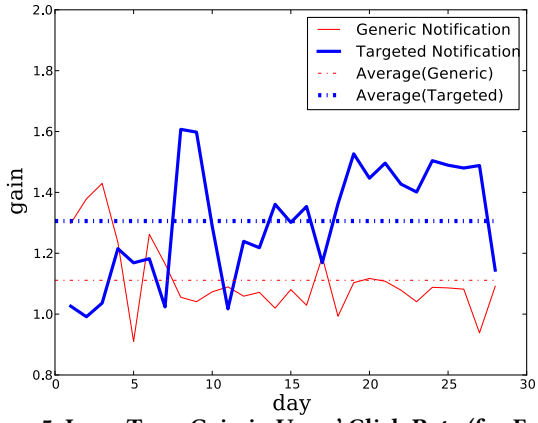

Figure 5: Long-Term Gain in Users' Click Rate (for Each Notification Class)

7.1 Long-Term Gain in Users' Click Rate

Figure 4 shows the gain in click rate for notifications, based on the comparisons of absolute click rates between the experimental group and the control group. The left side recaps the results in the previous study [26] while the right side shows the results from the current evaluation.

The highlight here is that, comparing both sides, our new system resulted in significantly more gain values. On the right side, over the entire study period (28 days), our system resulted in an average gain of 23.3%. The maximum daily gain during this study period was 41.6%. (A pair-wise t test revealed a significant effect of the notification strategy on clicks ($p < 0.01$).) Meanwhile, on the left hand side, the previous system resulted in 1.9% gain on average. Comparing the both average numbers, the new system achieved more than 10 times more gain. A Mann-Whitney test revealed a statistical significance between two results ($p < 0.001$).

This click gain of more than 20% (average) is actually surprisingly high, not only to us as researchers but also to the corporate marketing department. Yahoo! JAPAN's annual ad sales amount to 2.5 billion USD, with more than 59 million daily unique users (counted by number of browser instances) from smartphones. Even a 1% gain has a big influence on business that will be eventually reflected in more and better services offered to users in the future. In such a situation, the actual gains observed in this evaluation are definitely significant.

Finally, we also can observe certain variation of the gain over time. On this, we further discuss later in Section 9.

8 RESULT (2): TARGETED NOTIFICATION CONTENT

To investigate how well the system works with targeted notification content in addition to generic (common) content (Challenge (2)), we analyzed the long-term gain and short-term user response for those notification classes.

8.1 Long-Term Gain in Click Rate

Figure 5 shows the relative gain for the generic and targeted notification contents. As illustrated in the figure, gain was greater for the targeted notification class. The average gain throughout the entire evaluation period was 30.6% for the targeted class, while it was 11.1% for the generic class. The maximum daily gain of the targeted class was 60.7% on day 9, while that of the generic class was 42.9% on day 3. A t test confirmed statistically significant effects of the notification class ($p < 0.01$). From these results, We confirmed that the notifications with the personalized content experienced a significantly higher gain compared to those with the generic content.

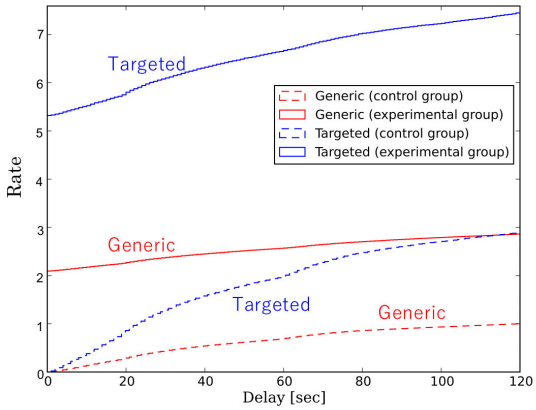

Figure 6: User Short-Term Response for Notification

8.2 Short-Term Response Behavior

Furthermore, we looked at the results on a finer scale. In the web marketing industry, it is often said that the first 60 seconds after notification delivery matter most in getting the attention of users. Here, we doubled the duration and investigated the first 120 seconds just in case.

Figure 6 shows, for both user groups and notification classes, the accumulative click rate with respect to the delay from when a notification was posted to Android to when a user clicked it. (Please note that, since the absolute click rate and number are confidential, the y-axis unit in the figure is a multiple whose base value 1 is the final click rate for the generic notification content in the control group.)

Firstly, comparing between both user groups, we see how the experimental group user quickly responds regardless of the notification content difference. Immediately after delivery (within 1 second), the click rates go up to more than 2 (generic content) and even up to more than 5 (targeted content) in the experimental group. (Note that since the measurement accuracy of the timestamp in our implementation is 1 second, all the clicks recorded within 1 second are plotted at time 0.) Meanwhile, in the control group (where notifications were delivered immediately), the click rate started from 0 and increased gradually for both notification classes. We believe this result explains why breakpoints are good timings for users to click incoming notifications.

Next, we compare the results of two different notification content classes. We see more click rate gain in the targeted notification contents. After 120 seconds of notification delivery, for the targeted content, the click rate in the experimental group resulted in 2.86 times higher than that in the control group. On the other hand, the corresponding number was 2.60 times in case of the generic content.

From these results, we confirmed that our breakpoint-based notification scheduling increases user's quick responses to both to both notification classes, especially the targeted notification class.

8.3 Delay in Notification Delivery Due to Breakpoint Detection

For the experimental group, delivery of notifications (after the client app received the content) was delayed for a certain period while the client looked for an immediate breakpoint. How long was that delay? We compare this delay for the two notification content cases.

Figure 7 illustrates the cumulative distribution function (CDF) of the delay from when a notification content arrived at the client to when the breakpoint logic detected a breakpoint and the notification was actually shown. The upper figure illustrates the results for two notification classes combined, while the lower figure illustrates the results for each of two classes.

Firstly, looking at the upper figure, the value converged to 1 at 3,600 seconds (1 hour), since we configured the timeout in breakpoint detection. 8.9% of the notifications were delivered within 1 minute, and 79.0% were delivered within 10 minutes. The overall average delay was 430.1 seconds (approximately 7 minutes). The timeout occurrence rate was 2.4%.

Looking at the lower figure for more detail for each notification class, we can see differences between the CDFs of the targeted and generic classes. In short, the generic notification class had a shorter delay after delivery. We consider that this occurred because the generic notifications were systematically scheduled to be delivered at four golden time slots (i.e. 8:00AM, noon, 6:00PM, and 9:00PM) when people's receptivity to information is considered to be high. On the other hand, the targeted class notifications were sent out from the server at manually chosen timings. Since the users' basic receptivity might be lower in some cases (e.g., when they are away from the phone for a long period), detecting the immediate breakpoint in the core estimation execution might take more time. We hypothesize that this difference caused the differences in the result.

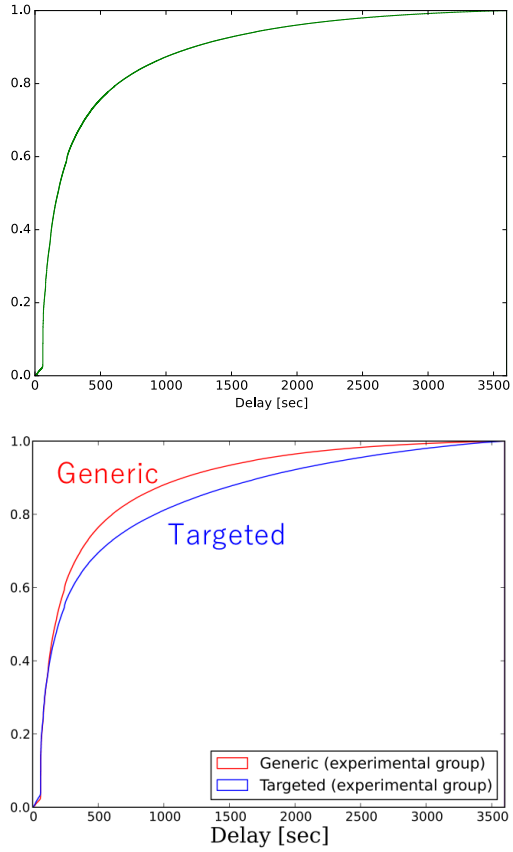

Figure 7: Delay in Notification Delivery

Table 4: Examples of Breaking News

Day	News Content
3	A missile was launched from a neighboring country.
3	The missile has passed through national airspace.
3	The missile has crashed into the sea.
13	Politician (A) launched a new party (X).
13	The prime minister dissolved the Lower House.
13	The prime minister held a press conference.

9 RESULT (3): BREAKING NEWS SITUATION

Next interesting finding is that our system actually gave a greater gain than at other times on a day when numerous breaking news push notifications were issued (challenge (3)). The breaking news on this occasion included a report on a big earthquake, big news in politics, and threats from a neighboring country (see Table 4) [3].

Figure 8 shows the relationship between the daily number of breaking news notifications (bar) and the click rate gain of the system (line). Furthermore, the green dots are the click rate gain of breaking news notifications. Note that these occasional breaking news notifications were delivered **immediately to both groups** because breakpoint-based scheduling was not enabled.

Looking at the line and the bars, we can see that the click rate gain in the experimental group tended to be higher on days when

[3]Note: The descriptions are not the exact text of the notifications.

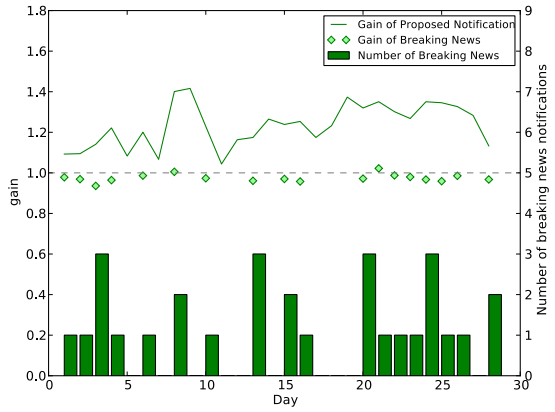

Figure 8: Number of Breaking News Notifications and Gain of Proposed System

one or more breaking news notifications were posted. We checked this by doing a t-test on the click rate gain between days with zero breaking news notifications and days with one or more breaking news notifications. The result showed $p = 0.06$. Although this value solely cannot confirm statistical significance, we believe that obtaining more data points would show even lower p value.

On the other hand, when we look at the bars and dots, very interestingly we can see that the click rate of posted mass breaking news notifications tend to be **lower in the experimental group** (i.e., the dot values under 1.0) compared with the control group. We found statistical significance ($p < 0.001$) on this from a t-test on the click rate of breaking news notifications.

Although further investigation is needed, our hypothesis on this interesting result is as follows. Since the breakpoint-scheduled notifications (of the "Recommend" and "Targeted" classes) were delivered at the user's opportune timings, the users were considered to have clicked and accessed such notifications (this conclusion is supported by the results in Section 8.2). Hence, accesses to the breaking news notifications, which had been delivered at non-breakpoint timings, might have been ignored more often.

We also need to pay attention to the fact that the number of breaking news notifications (1 to 3) is significantly smaller than other random notifications (about up to 100 notifications a day according to [21]) from random applications in random (non-breakpoint) timings. Why did an extra 1-3 notifications cause this observable difference in gain? Our hypothesis is as follows. Thanks to the effectiveness of our proposed system, whether any breaking news come or not, the experimental group users anyway accessed to our notification content ("Recommend" or "Targeted" classes) presented in his/her breakpoint timings. However, the control group users were considered to access more breaking news contents than our notification content, in the days when any breaking news occurred. Thus, for them, the amount of access to our notifications actually decreased. In this way, comparing the access between two user groups, the gain were considered to be higher in the days with breaking news notifications.

10 DISCUSSION

From the deployment study of our new adaptive notification scheduling system on the production environment with 382,518 users with 28 days, we confirmed significant results including significant improvement in click rate gain, the fact that our adaptive notification scheduling performs better for targeted notification contents, and very interesting outcome that our method works even better in situations with breaking news notifications. **From these promising results, we decided to enable the proposed mechanism to the entire user base of our service, as a permanent component. Now the proposed system is used by more than 10 million Android end users.**

In spite of that, we still see some possible future research opportunities. We need further investigations regarding the interesting finding on the relationship between breakpoint-based notification and the breaking news. Generally speaking, breaking news contains important news for all users. Thus, any reduction in its click rate should be avoided. However, at the same time, this finding revealed the effectiveness of our system from yet another viewpoint.

We are also interested in system (device) wide deployment of our approach inside operating systems and/or collaboration between multiple applications implementing our approach. This effort would involve system-wide optimization (sensing and classification of breakpoints) and user-centric optimization of notification delivery.

Placement of the interruptibility estimation model between the client and server sides depending on the context and resource availability is another fascinating future challenge. Depending on the strong features in the classifier model (e.g., time of the day), the system can be optimized by executing a certain portion of the classifier calculation on the server to avoid sending notifications that eventually activate the client-side classification.

Use of notification-content-related features to the client-side interruptibility classifier would be another challenging venue. Our breakpoint-based interruptibility estimation approach (with a generic classifier model common for all users) basically based on the previous finding that, common to many people, a breakpoint is generally a good timing for information consumption. Assuming that people's primary task content and/or the preference on the notification content are versatile for each user, we hypothesize that introducing such features will need an introduction of personalized classifier for each user. We also estimate that such revision on the system is quite costly in our case. In this research, since we particularly wanted to focus on how well the system with a generic model behaves with personalized notifications (without introducing such a costly revision), we did not use such features.

11 CONCLUSION

Mobile users are facing an "interruption overload" problem with the increasing use of push information delivery. Towards permanent production deployment of mobile sensing and ML-based interruptibility estimation and corresponding adaptive notification scheduling, our focus in this work were on the basic performance in click rate gain on the product environment, as well as handling configurations and situations like personalized notification contents and additional mass notification situations. From the promising results of our study with approaches spanning from the system, deployment, up to the evaluation layers, we conclude that our system works with significant better impact on notification performance which is sufficient for the permanent deployment for all the users.

REFERENCES

[1] [n. d.]. Attention Management in Ubiquitous Computing Environments (AMUCE 07). http://www.ubicomp.org/ubicomp2007/index-14.htm.

[2] Utku Acer, Afra Mashhadi, Claudio Forlivesi, and Fahim Kawsar. 2015. Energy Efficient Scheduling for Mobile Push Notifications. In *Proceedings of the 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services on 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MOBIQUITOUS'15)*. 100–109. https://doi.org/10.4108/eai.22-7-2015.2260067

[3] Piotr D. Adamczyk and Brian P. Bailey. 2004. If not now, when?: the effects of interruption at different moments within task execution. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. 271–278. https://doi.org/10.1145/985692.985727

[4] Brian P. Bailey and Joseph A. Konstan. 2006. On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state. *Computers in Human Behavior* 22, 4 (2006), 685 – 708. https://doi.org/10.1016/j.chb.2005.12.009

[5] James "Bo" Begole, Nicholas E. Matsakis, and John C. Tang. 2004. Lilsys: Sensing Unavailability. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW '04)*. 511–514. https://doi.org/10.1145/1031607.1031691

[6] Mary Czerwinski, Edward Cutrell, and Eric Horvitz. 2000. Instant messaging: Effects of relevance and timing. In *People and computers XIV: Proceedings of HCI*, Vol. 2. British Computer Society, 71–76.

[7] Joel E. Fischer, Chris Greenhalgh, and Steve Benford. 2011. Investigating Episodes of Mobile Phone Activity As Indicators of Opportune Moments to Deliver Notifications. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11)*. 181–190. https://doi.org/10.1145/2037373.2037402

[8] D. Garlan, D.P. Siewiorek, A. Smailagic, and P. Steenkiste. 2002. Project Aura: toward distraction-free pervasive computing. *Pervasive Computing, IEEE* 1, 2 (april-june 2002), 22 –31. https://doi.org/10.1109/MPRV.2002.1012334

[9] Google Inc. [n. d.]. Making Your App Location-Aware - Android Developers. https://developer.android.com/intl/ja/training/location/index.html.

[10] Sandy Gould, Duncan Brumby, Anna Cox, Victor González, Dario Salvucci, and Niels Taatgen. 2012. Multitasking and interruptions: a SIG on bridging the gap between research on the micro and macro worlds. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*. 1189–1192.

[11] Joyce Ho and Stephen S. Intille. 2005. Using Context-aware Computing to Reduce the Perceived Burden of Interruptions from Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. 909–918. https://doi.org/10.1145/1054972.1055100

[12] Eric Horvitz and Johnson Apacible. 2003. Learning and Reasoning About Interruption. In *Proceedings of the 5th International Conference on Multimodal Interfaces (ICMI '03)*. 20–27. https://doi.org/10.1145/958432.958440

[13] Eric Horvitz, Paul Koch, and Johnson Apacible. 2004. BusyBody: Creating and Fielding Personalized Models of the Cost of Interruption. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW '04)*. 507–510. https://doi.org/10.1145/1031607.1031690

[14] Scott Hudson, James Fogarty, Christopher Atkeson, Daniel Avrahami, Jodi Forlizzi, Sara Kiesler, Johnny Lee, and Jie Yang. 2003. Predicting Human Interruptibility with Sensors: A Wizard of Oz Feasibility Study. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. 257–264. https://doi.org/10.1145/642611.642657

[15] Shamsi T. Iqbal and Brian P. Bailey. 2006. Leveraging Characteristics of Task Structure to Predict the Cost of Interruption. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. 741–750. https://doi.org/10.1145/1124772.1124882

[16] Shamsi T. Iqbal and Brian P. Bailey. 2005. Investigating the Effectiveness of Mental Workload As a Predictor of Opportune Moments for Interruption. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems (CHI EA '05)*. 1489–1492. https://doi.org/10.1145/1056808.1056948

[17] Shamsi T. Iqbal and Brian P. Bailey. 2010. Oasis: A Framework for Linking Notification Delivery to the Perceptual Structure of Goal-directed Tasks. *ACM Transactions on Computer-Human Interaction* 17, 4, Article 15 (Dec. 2010), 28 pages. https://doi.org/10.1145/1879831.1879833

[18] Shamsi T. Iqbal and Eric Horvitz. 2010. Notifications and Awareness: A Field Study of Alert Usage and Preferences. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work (CSCW '10)*. 27–30. https://doi.org/10.1145/1718918.1718926

[19] Daniel Kahneman. 1973. *Attention and effort*. Prentice-Hall, Inc.

[20] J. G. Kreifeldt and M. E. McCarthy. 1981. Interruption as a test of the user-computer interface. In *JPL Proceeding of the 17 th Annual Conference on Manual Control*. 655–667.

[21] Abhinav Mehrotra, Mirco Musolesi, Robert Hendley, and Veljko Pejovic. 2015. Designing Content-driven Intelligent Notification Mechanisms for Mobile Applications. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. 813–824. https://doi.org/10.1145/2750858.2807544

[22] James T Milord and Raymond P Perry. 1977. A Methodological Study of Overloadx. *The Journal of General Psychology* 97, 1 (1977), 131–137.

[23] Darren Newtson and Gretchen Engquist. 1976. The perceptual organization of ongoing behavior. *Journal of Experimental Social Psychology* 12, 5 (1976), 436–450.

[24] Tadashi Okoshi, Julian Ramos, Hiroki Nozaki, Jin Nakazawa, Anind K. Dey, and Hideyuki Tokuda. 2015. Attelia: Reducing User's Cognitive Load due to Interruptive Notifications on Smart Phones. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications 2015 (PerCom '15)*.

[25] Tadashi Okoshi, Julian Ramos, Hiroki Nozaki, Jin Nakazawa, Anind K. Dey, and Hideyuki Tokuda. 2015. Reducing Users' Perceived Mental Effort Due to Interruptive Notifications in Multi-device Mobile Environments. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. 475–486.

[26] Tadashi Okoshi, Kota Tsubouchi, Masaya Taji, Takanori Ichikawa, and Hideyuki Tokuda. 2017. Attention and Engagement-Awareness in the Wild : A Large-Scale Study with Adaptive Notifications. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications 2017 (PerCom '17)*. 100–110.

[27] Veljko Pejovic and Mirco Musolesi. 2014. InterruptMe : Designing Intelligent Prompting Mechanisms for Pervasive Applications. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*. 395–906. https://doi.org/10.1145/2493432.2493445

[28] Martin Pielot, Tilman Dingler, Jose San Pedro, and Nuria Oliver. 2015. When Attention is Not Scarce - Detecting Boredom from Mobile Phone Usage. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. 825–836.

[29] Benjamin Poppinga, Martin Pielot, Niels Henze, Nuria Oliver, Karen Church, and Alireza Sahami Shirazi. 2015. Smarttention, Please! Intelligent Attention Management on Mobile Devices. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI '15)*. 1066–1069. https://doi.org/10.1145/2786567.2795399

[30] Richard M Shiffrin and Walter Schneider. 1977. Controlled and automatic human information processing: II. Perceptual learning, automatic attending and a general theory. *Psychological review* 84, 2 (1977), 127.

[31] Herbert A Simon. 1971. Designing organizations for an information-rich world. *Computers, communication, and the public interest* 37 (1971), 40–41.

[32] Cheri Speier, Joseph S Valacich, and Iris Vessey. 1999. The influence of task interruption on individual decision making: An information overload perspective. *Decision Sciences* 30, 2 (1999), 337–360.

[33] Robert J. Sternberg and Karin Sternberg. 2012. *Cognitive Psychology* (6 ed.). Cengage Learning.

[34] G. H. (Henri) ter Hofte. 2007. Xensible Interruptions from Your Mobile Phone. In *Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '07)*. 178–181. https://doi.org/10.1145/1377999.1378003

[35] Alvin Toffler. 1990. *Future shock*. Bantam.

[36] Kota Tsubouchi and Tadashi Okoshi. 2017. People's Interruptibility In-the-wild: Analysis of Breakpoint Detection Model in a Large-scale Study. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers (UbiComp '17)*. ACM, New York, NY, USA, 922–927. https://doi.org/10.1145/3123024.3124556

[37] Alexandra Voit, Benjamin Poppinga, Dominik Weber, Matthias BoĬLhmer, Niels Henze, Sven Gehring, Tadashi Okoshi, and Veljko Pejovic. 2016. UbiTtention: Smart & Ambient Notification and Attention Management. In *Adjunct Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16)*. 1520–1523.

[38] Dominik Weber, Alireza Sahami Shirazi, Sven Gehring, Niels Henze, Benjamin Poppinga, Martin Pielot, and Tadashi Okoshi. 2016. Smarttention, Please!: 2Nd Workshop on Intelligent Attention Management on Mobile Devices. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI '16)*. 914–917. https://doi.org/10.1145/2957265.2965025

[39] Fred RH Zijlstra, Robert A Roe, Anna B Leonora, and Irene Krediet. 1999. Temporal factors in mental work: Effects of interrupted activities. *Journal of Occupational and Organizational Psychology* 72, 2 (1999), 163–185.